VHDL Cheat Sheet – Combinational Logic

Libraries

- ALWAYS start each VHDL file with the following 2 lines

```
library ieee;
use ieee.std_logic_1164.all;
```

Data Types

- Two most common: std_logic, std_logic_vector(<N> downto 0)

Entities

- Defines a black-box interface for your module

```
entity MyEntity is
      port ( a, b : in std_logic;
                  c : out std_logic);
end entity;
```

Architectures

- Defines your implementation of your entity
- Include any signal declarations before 'begin'

```
architecture main of MyEntity is
      signal <blah>;
begin
      <blah>
end architecture;
```

Component Declaration

- Any architectures that use an external component must have a component declaration

```
architecture main of MyEntity is
      component DFF
      port( d : in std_logic;
                  q : out std_logic);
      end component;
      signal EntityD, EntityQ : std_logic;
begin
      DFF_1 : DFF
            Port map (d => EntityD, q => EntityQ);
end architecture;
```

Concurrent Assignment

- Describes a wired connection

```
a <= b and c;
```

The following 2 can only be written OUTSIDE a process

Selected Assignment

```
With Sel select
      Y <=  A when "00",
```

```
            B when "01",
            C when "10",
            D when others;
```

Conditional Assignment

```
Y <= A when Sel = "00" else
      B when Sel = "01" else
      C when Sel = "10" else
      D when others;
```

Combinational Processes

- Another way to describe combinational logic
- A process is combinational if it doesn't infer any registers

```
process(<sensitivity list>)
begin
      <blah>
end process;
```

The following 2 can only be written INSIDE a process

If-Then-Else

```
process(a)
begin
      if a = '0' then
            q <= '0'
      else
            q <= '1'
      end if;
end process;
```

Case Statement

```
process(a)
begin
      case a
            when '0' => q <= '0';
            when others => q <= '1';
      end case;
end process;
```

Sequential Processes

- Describes sequential logic
- A process is sequential if it infers at least one register

```
process(clk)
begin
      if rising_edge(clk) then
            <blah>
      end if;
end process;
```